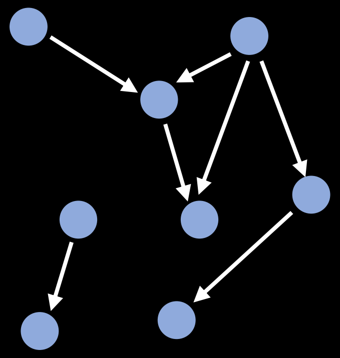
A graph consists of a set of vertices V, and a set of edges E.

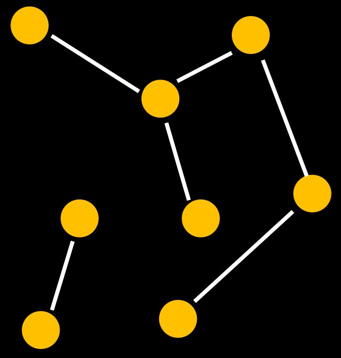
directed graphs



$E \subseteq V \times V$

The edge from v to w is written $v \to w$

undirected graphs

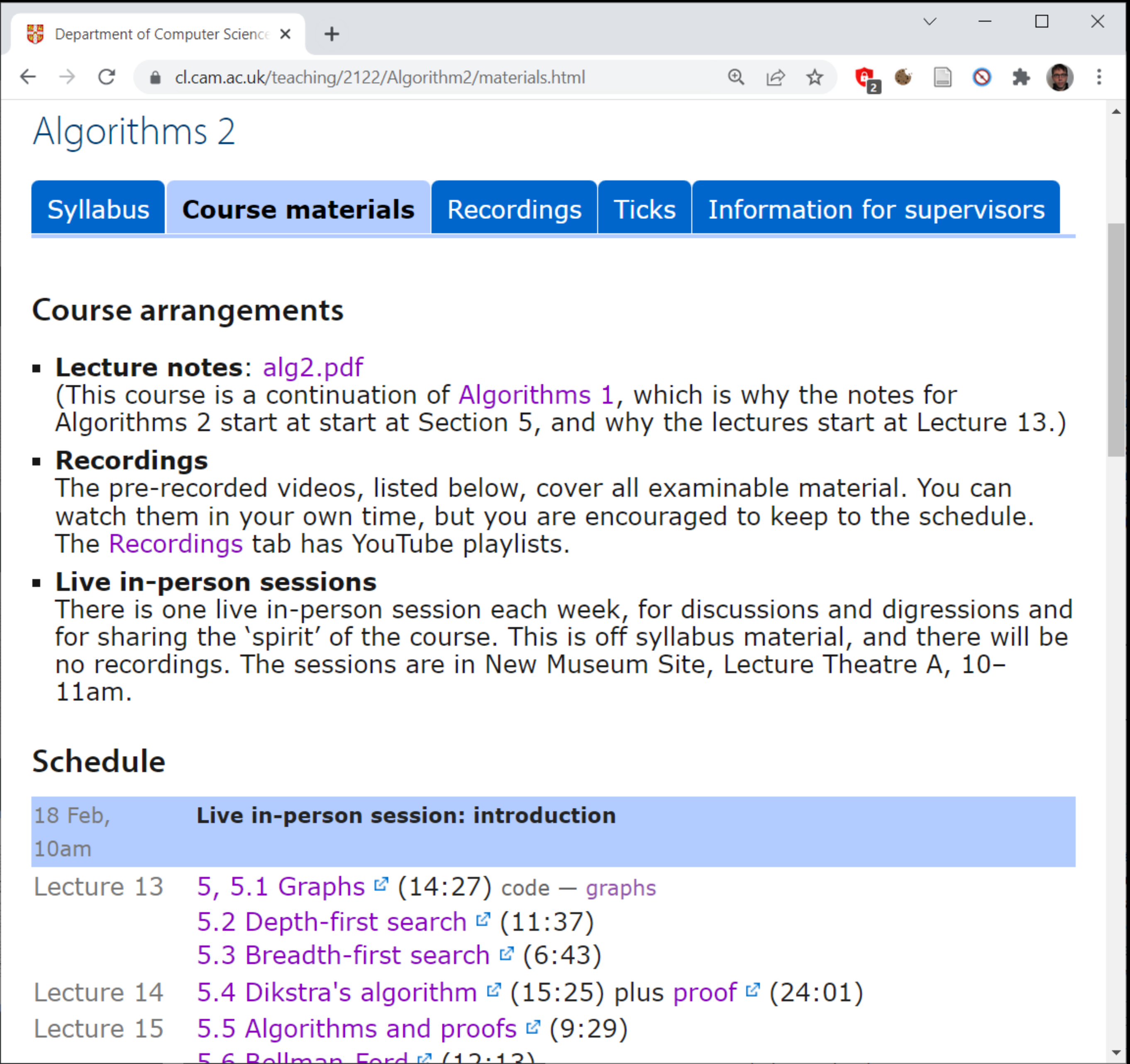


$E \subseteq$ subsets of V of size 2

The edge between v and w is written $v \leftrightarrow w$

… but you'll learn all this from the videos and notes, and the live in-person sessions are for wider-ranging discussion.

Arrangements



Department of Computer Science
cl.cam.ac.uk/teaching/2122/Algorithm2/materials.html

Algorithms 2

Syllabus | **Course materials** | Recordings | Ticks | Information for supervisors

Course arrangements

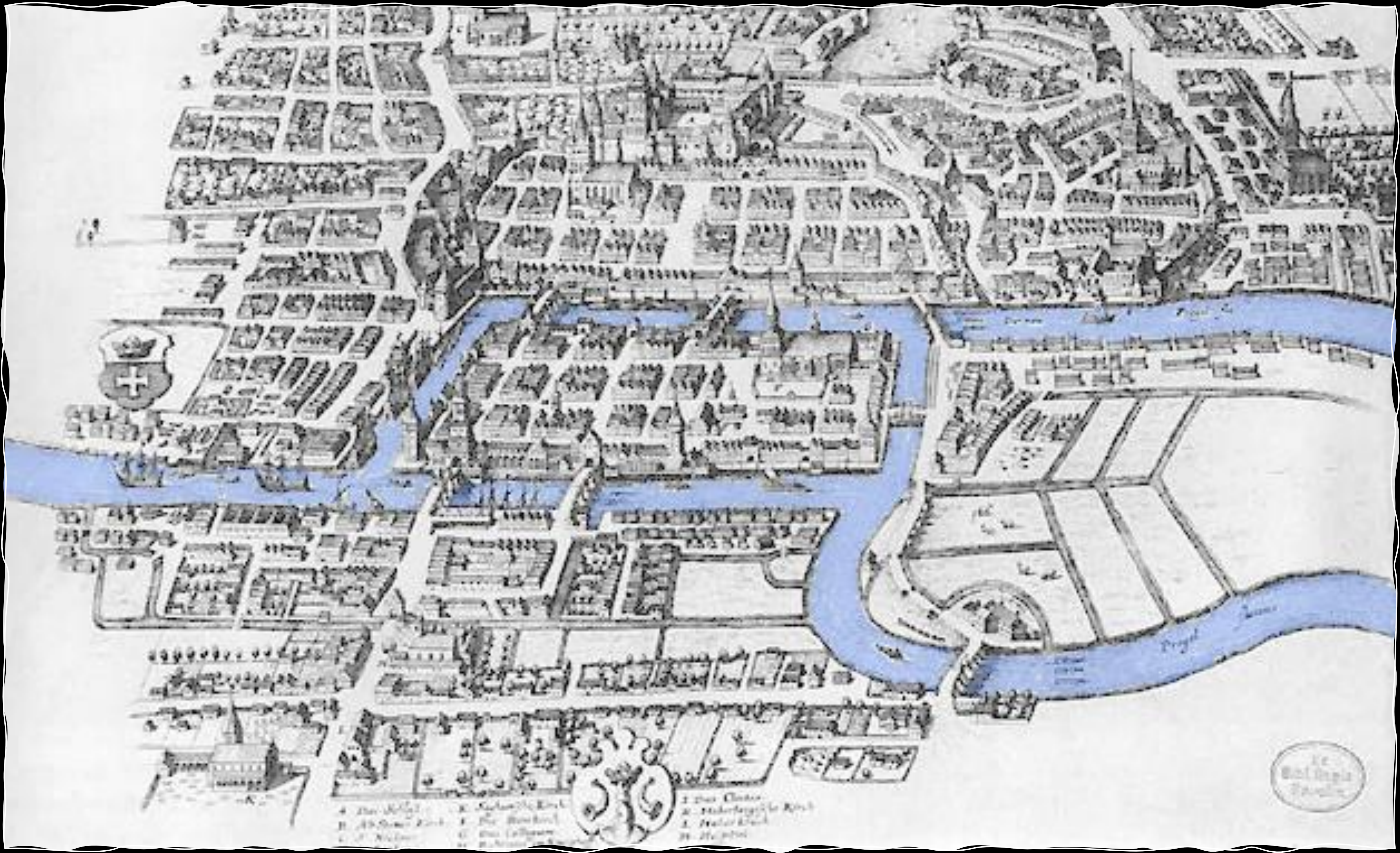
- **Lecture notes**: alg2.pdf
  (This course is a continuation of Algorithms 1, which is why the notes for Algorithms 2 start at start at Section 5, and why the lectures start at Lecture 13.)
- **Recordings**
  The pre-recorded videos, listed below, cover all examinable material. You can watch them in your own time, but you are encouraged to keep to the schedule. The Recordings tab has YouTube playlists.
- **Live in-person sessions**
  There is one live in-person session each week, for discussions and digressions and for sharing the 'spirit' of the course. This is off syllabus material, and there will be no recordings. The sessions are in New Museum Site, Lecture Theatre A, 10–11am.

Schedule

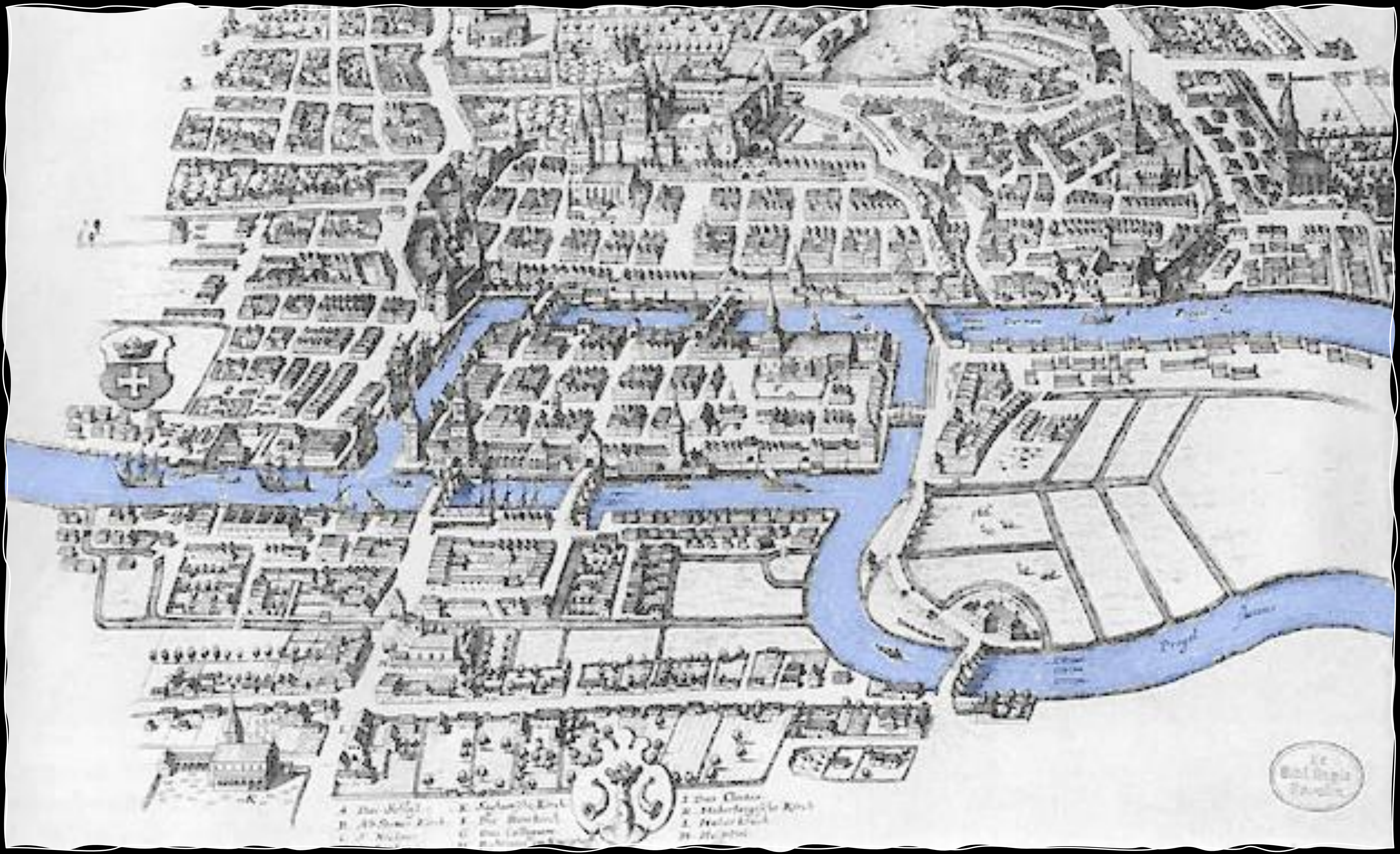
18 Feb, 10am	**Live in-person session: introduction**
Lecture 13	5, 5.1 Graphs (14:27) code — graphs 5.2 Depth-first search (11:37) 5.3 Breadth-first search (6:43)
Lecture 14	5.4 Dikstra's algorithm (15:25) plus proof (24:01)
Lecture 15	5.5 Algorithms and proofs (9:29) 5.6 Bellman-Ford (12:13)

- Pre-recorded videos and printed notes
  cover the examinable material
- Weekly live in-person sessions
  not recorded or streamed; interactive; non-examinable
- One required tick, several optional ticks
  (released next week)

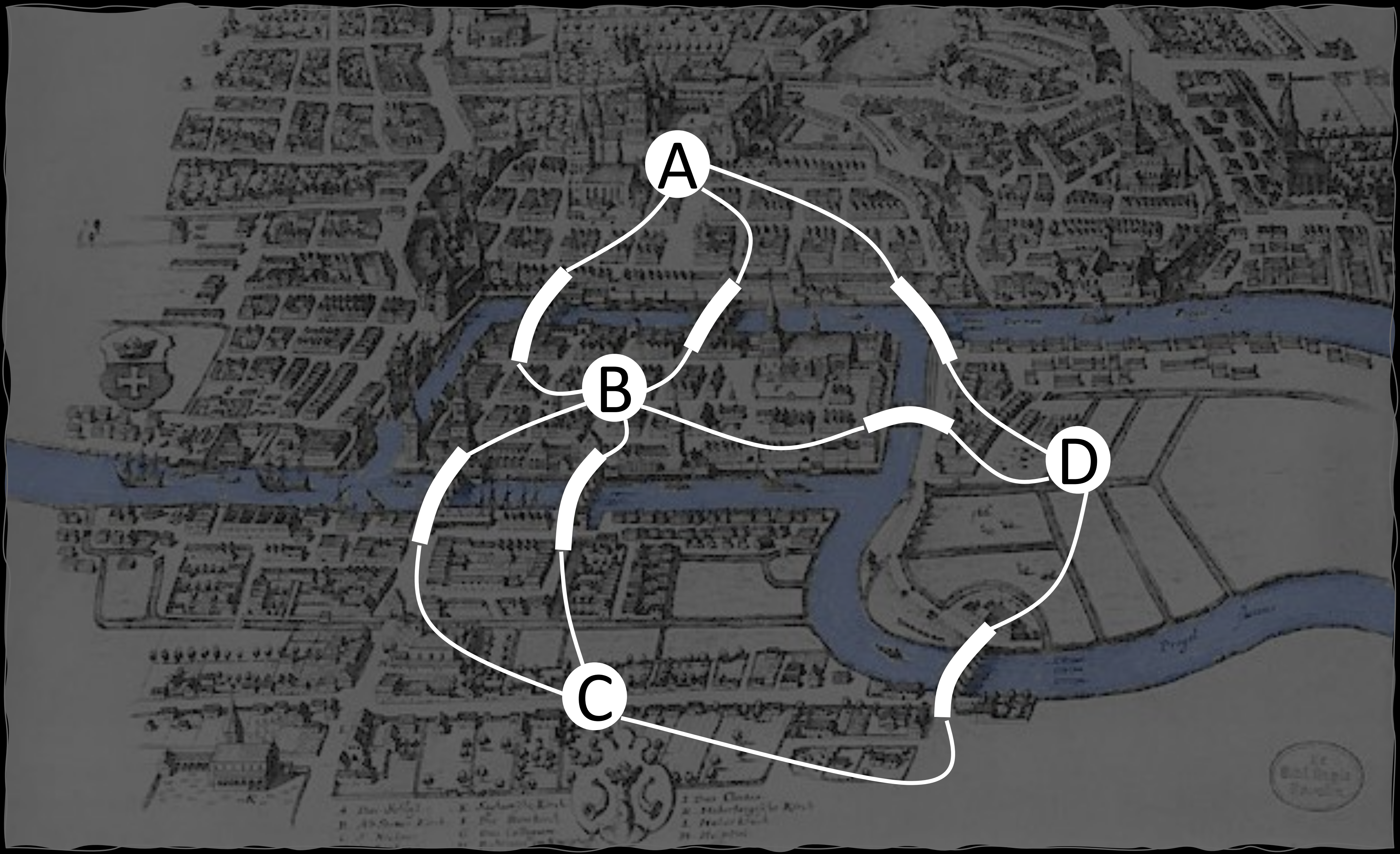
Konigsberga



"Can I go for a stroll around the city on a route that crosses each bridge exactly once?"

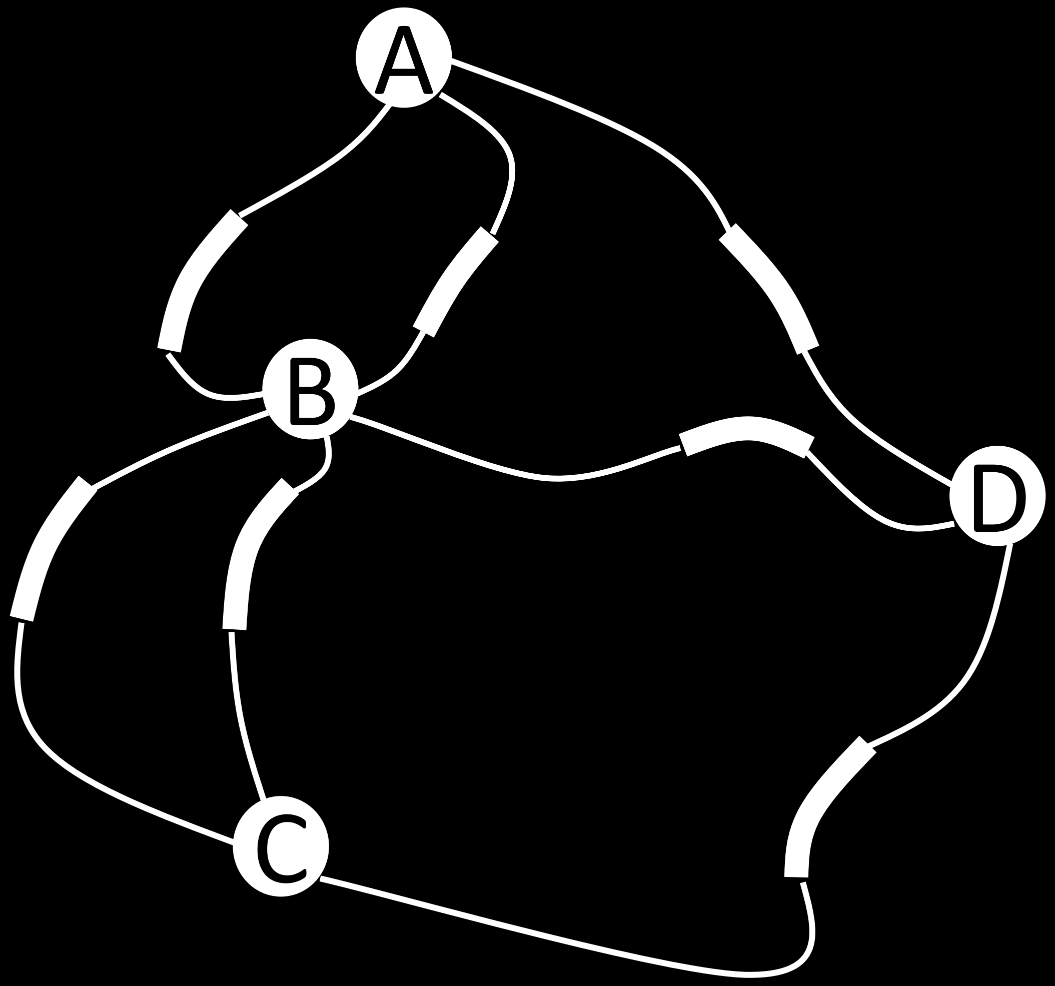


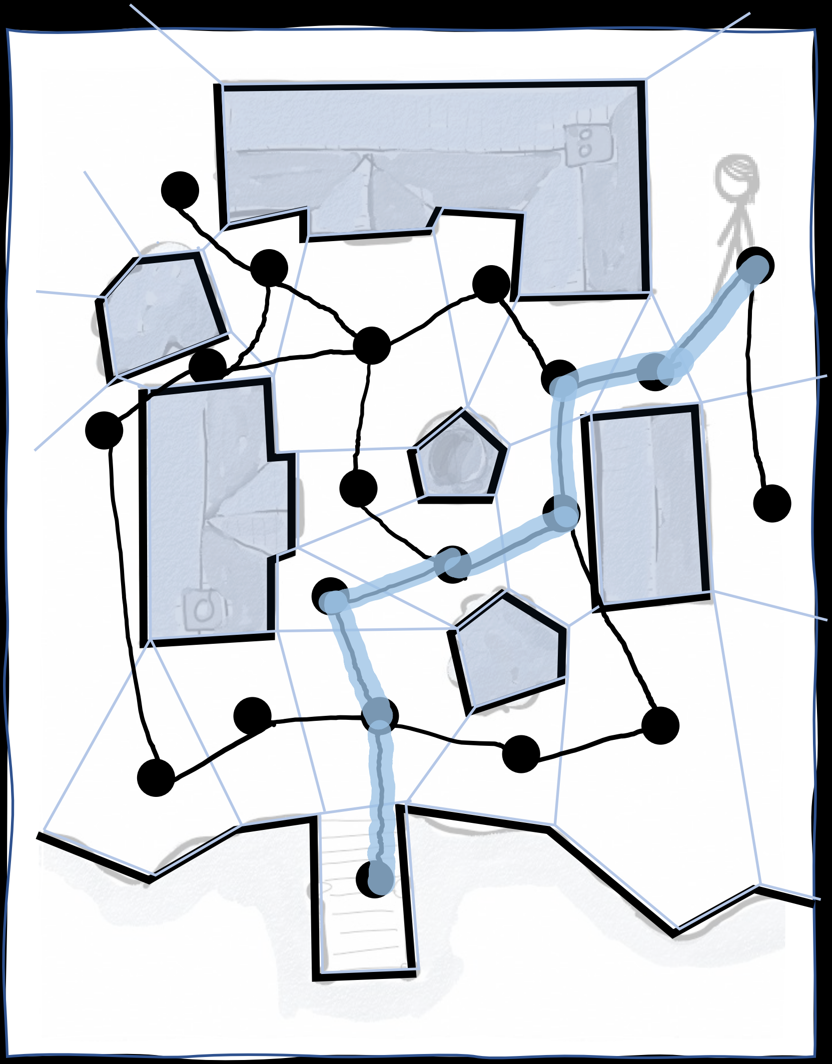
"Can I go for a stroll around the city on a route that crosses each bridge exactly once?"



“Is there a path in which every edge appears exactly once?”

```
g = {A: [B,B,D],
     B: [A,A,C,C,D],
     C: [B,B,D],
     D: [A,B,C]}
```





PATH-FINDING ALGORITHMS

How should this game agent navigate to the jetty?

1. Draw polygon boundaries around obstacles
2. Divide free space into convex polygons
3. Create a graph, with edges between adjacent polygons
4. Find a path on the graph
5. Draw this path in 2D coordinates on the map (easy, since we’ve used convex polygons)

Dwarf Fortress

Q: I’ve seen other games similar to Dwarf Fortress die on their pathfinding algorithms. What do you use and how do you keep it efficient?

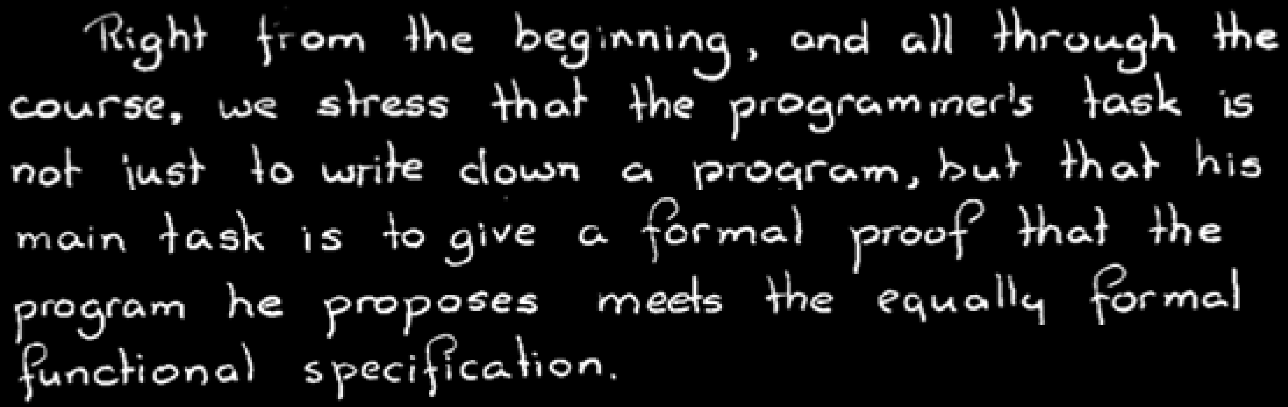
A: Yeah, the base algorithm is only part of it. We use A*, which is fast of course, but it’s not good enough by itself.

Generally, people have used approaches that add various larger structures on top of the map to cut corners. But we can’t take advantage of these innovations since our map changes so much.

Interview with Tarn Adams (developer) by Ryan Donovan from the StackOverflow blog, Dec 2021

What this course is about

- Clever algorithms
- Performance analysis
- Proving correctness

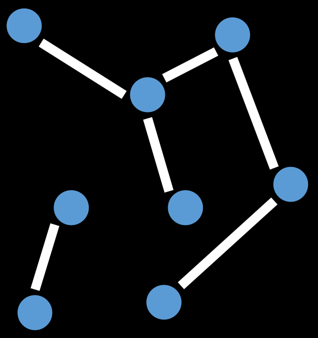


Right from the beginning, and all through the course, we stress that the programmer's task is not just to write down a program, but that his main task is to give a formal proof that the program he proposes meets the equally formal functional specification.

Edsger Dijkstra (1930—2002)
On the cruelty of really teaching computer science, 1988

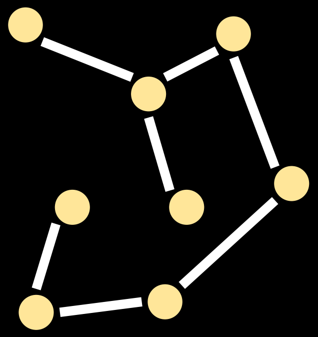
- A *cycle* is a path $v_0 \leftrightarrow v_1 \leftrightarrow \cdots \leftrightarrow v_k$ with at least two vertices, where $v_0 = v_k$
- A graph is *connected* if for every pair of vertices there is a path between them
- A *forest* is an undirected graph without any cycles
- A *tree* is a connected forest

$A.$



forest

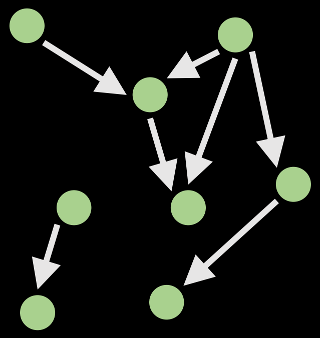
$B.$



tree

forest

$C.$



Which is a tree, which is a forest?

What this course is about

- Clever algorithms
- Performance analysis
- Proving correctness

- Interplay between data and algorithm

Dwarf Fortress

Q: I've seen other games similar to Dwarf Fortress die on their pathfinding algorithms. What do you use and how do you keep it efficient?

A: Yeah, the base algorithm is only part of it. We use A*, which is fast of course, but it's not good enough by itself.

Generally, people have used approaches that add various larger structures on top of the map to cut corners. But we can't take advantage of these innovations since our map changes so much.

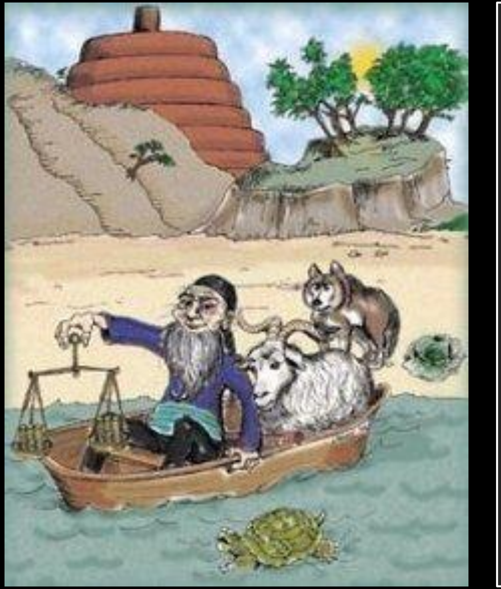
Interview with Tarn Adams (developer) by Ryan Donovan from the StackOverflow blog, Dec 2021

Q. If you were Google Maps,
what would you be trying to do?

What this course is about

- Clever algorithms
- Performance analysis
- Proving correctness

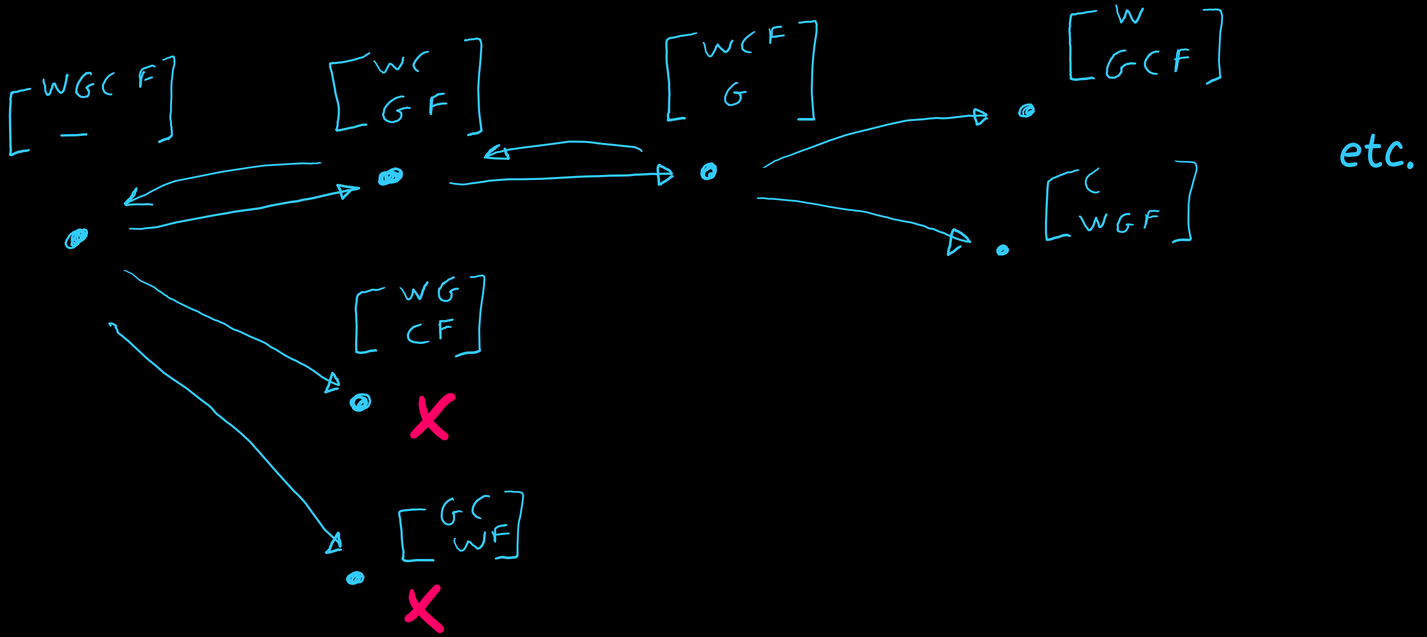
- Interplay between data and algorithm
- What we can model with graphs



Once upon a time a farmer went to a market and purchased a wolf, a goat, and a cabbage. On the way home, the farmer came to the bank of a river and rented a boat.

The boat can carry the farmer, plus a single one of the purchases. If the wolf and goat are left unattended together, the wolf will eat the goat. Likewise, the goat and the cabbage.

How should the farmer cross the river?



GAME-PLAY PROBLEMS

Let V be the set of possible game states.

Let there be an edge $v \to w$ if there is an action that transitions from v to w.

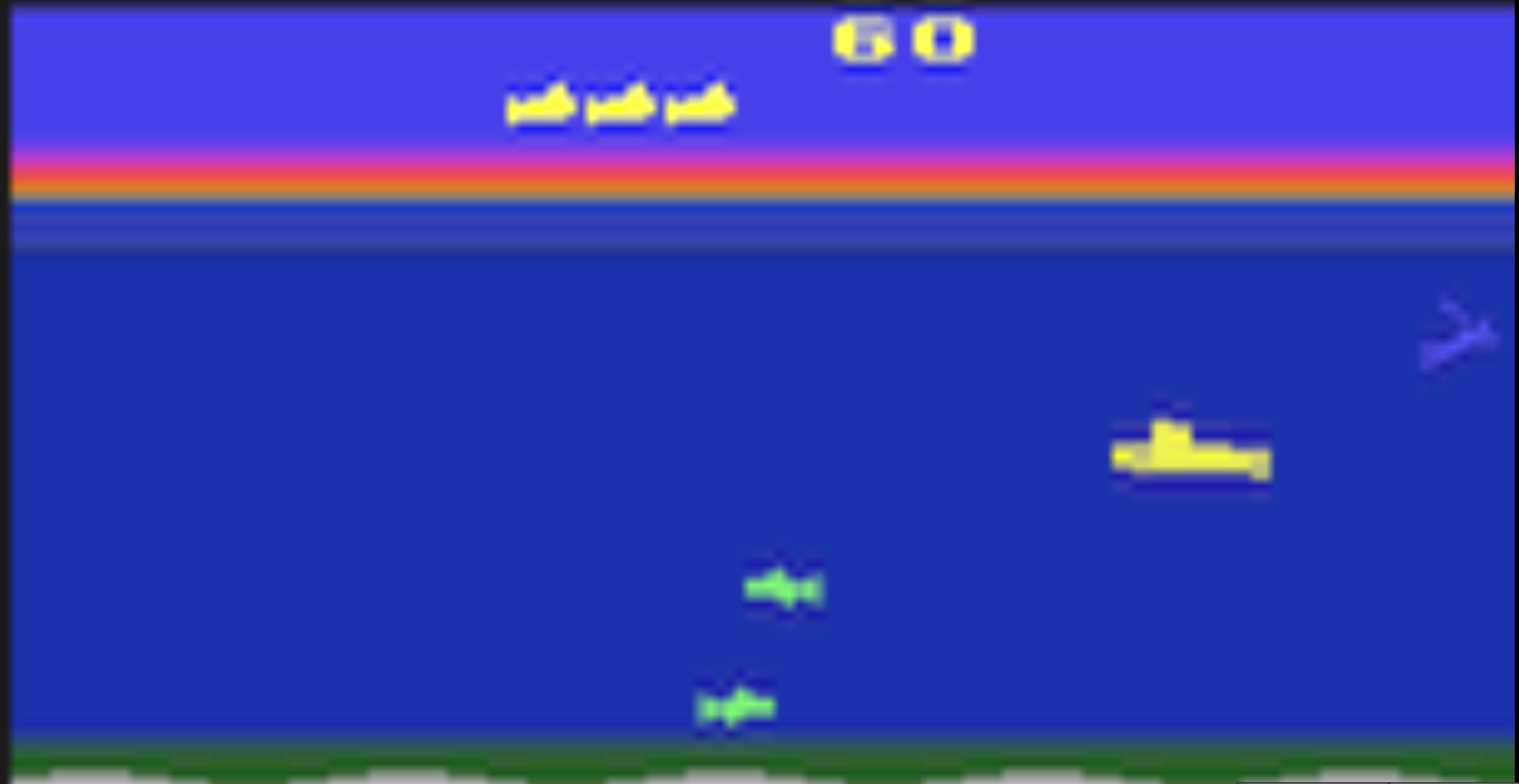
What is the shortest path from the initial state to the desired end-state?

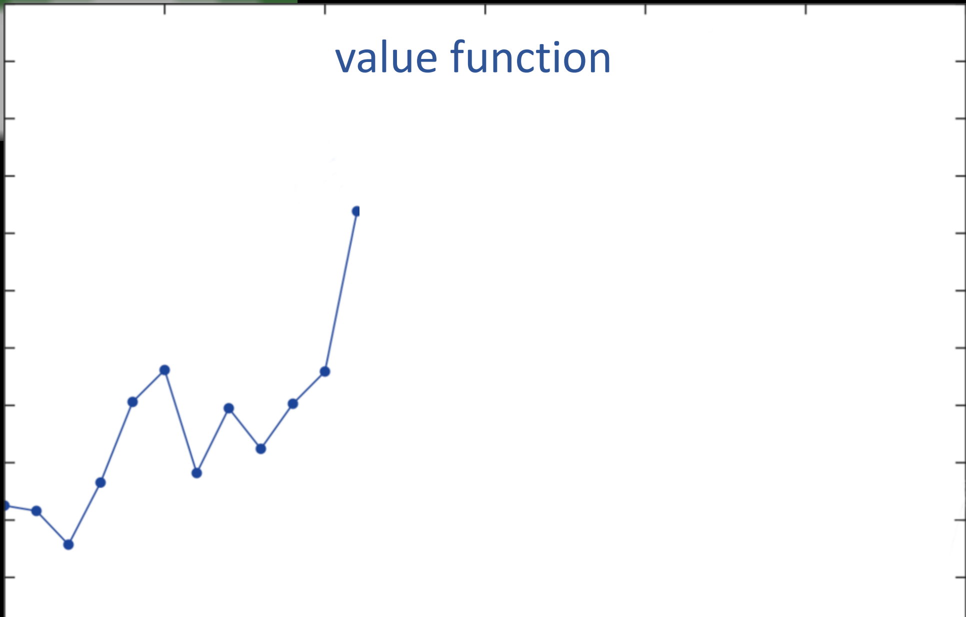
How can I train a neural network to play the game,
i.e. to pick the next action along the path from any starting point?

Training goal: learn a *value function* $F : V \to \mathbb{R}$ representing “how much I’ll win, starting from a given state”.

Gameplay: from any state v, simply pick next state

$$v_{\text{next}} = \underset{w\,:\,v \to w}{\arg\max}\, F(w)$$

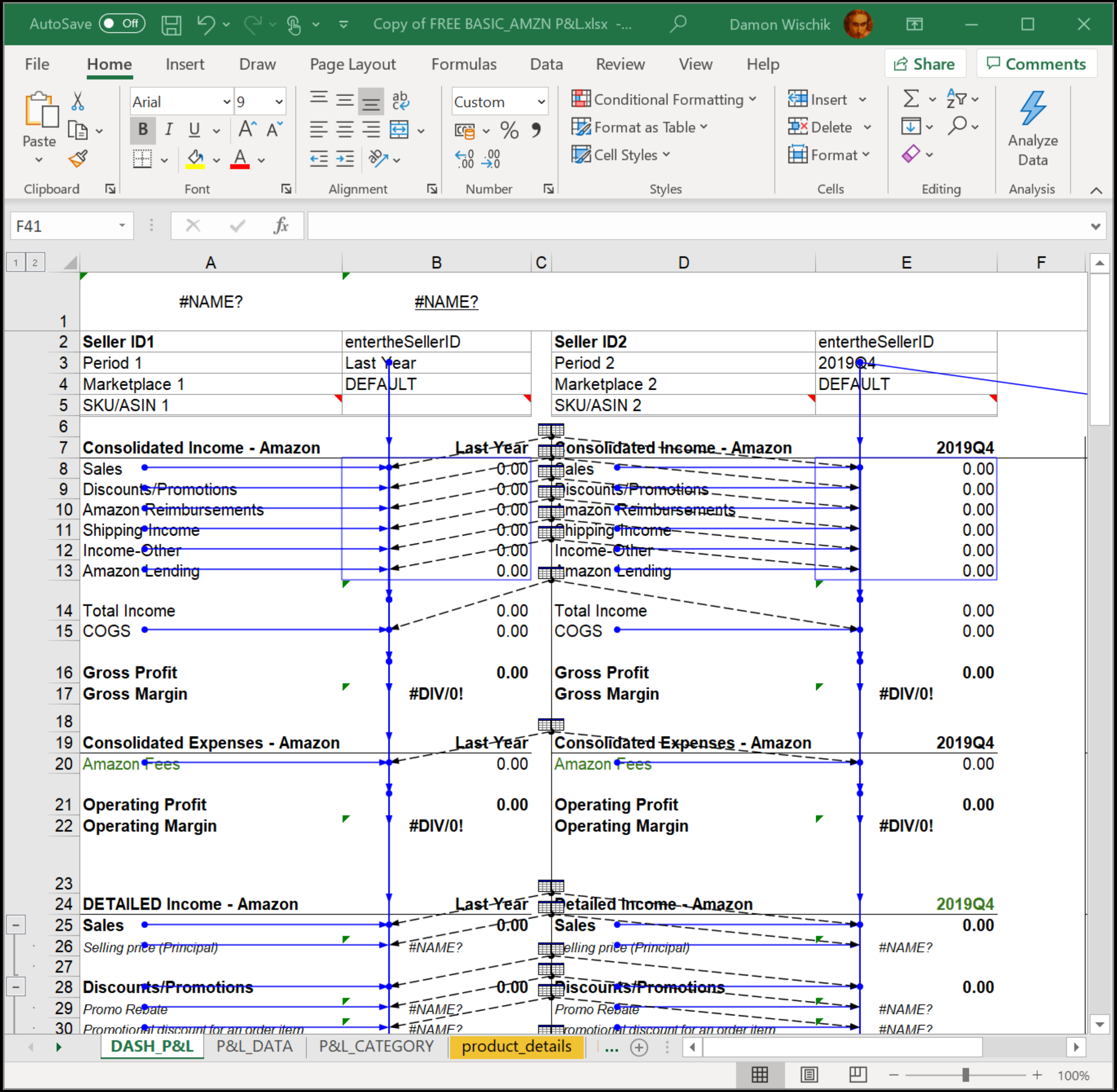




Training goal: learn a *value function* $F: V \to \mathbb{R}$ representing “how much I’ll win, starting from a given state”.

Gameplay: from any state v, simply pick next state

$$v_{\text{next}} = \underset{w\,:\,v\to w}{\arg\max}\, F(w)$$



	A	B	C	D	E	F
1	#NAME?	#NAME?				
2	**Seller ID1**	entertheSellerID		**Seller ID2**	entertheSellerID	
3	Period 1	Last Year		Period 2	2019Q4	
4	Marketplace 1	DEFAULT		Marketplace 2	DEFAULT	
5	SKU/ASIN 1			SKU/ASIN 2		
6						
7	**Consolidated Income - Amazon**	**Last Year**		**Consolidated Income - Amazon**	**2019Q4**	
8	Sales	0.00		Sales	0.00	
9	Discounts/Promotions	0.00		Discounts/Promotions	0.00	
10	Amazon Reimbursements	0.00		Amazon Reimbursements	0.00	
11	Shipping Income	0.00		Shipping Income	0.00	
12	Income-Other	0.00		Income-Other	0.00	
13	Amazon Lending	0.00		Amazon Lending	0.00	
14	Total Income	0.00		Total Income	0.00	
15	COGS	0.00		COGS	0.00	
16	**Gross Profit**	**0.00**		**Gross Profit**	**0.00**	
17	**Gross Margin**	**#DIV/0!**		**Gross Margin**	**#DIV/0!**	
18						
19	**Consolidated Expenses - Amazon**	**Last Year**		**Consolidated Expenses - Amazon**	**2019Q4**	
20	Amazon Fees	0.00		Amazon Fees	0.00	
21	**Operating Profit**	**0.00**		**Operating Profit**	**0.00**	
22	**Operating Margin**	**#DIV/0!**		**Operating Margin**	**#DIV/0!**	
23						
24	**DETAILED Income - Amazon**	**Last Year**		**Detailed Income - Amazon**	**2019Q4**	
25	**Sales**	**0.00**		**Sales**	**0.00**	
26	*Selling price (Principal)*	*#NAME?*		*Selling price (Principal)*	*#NAME?*	
27						
28	**Discounts/Promotions**	**0.00**		**Discounts/Promotions**	**0.00**	
29	*Promo Rebate*	*#NAME?*		*Promo Rebate*	*#NAME?*	
30	*Promotional discount for an order item*	*#NAME?*		*Promotional discount for an order item*	*#NAME?*	

What order should we compute the cells in a spreadsheet?

Is it even computable?

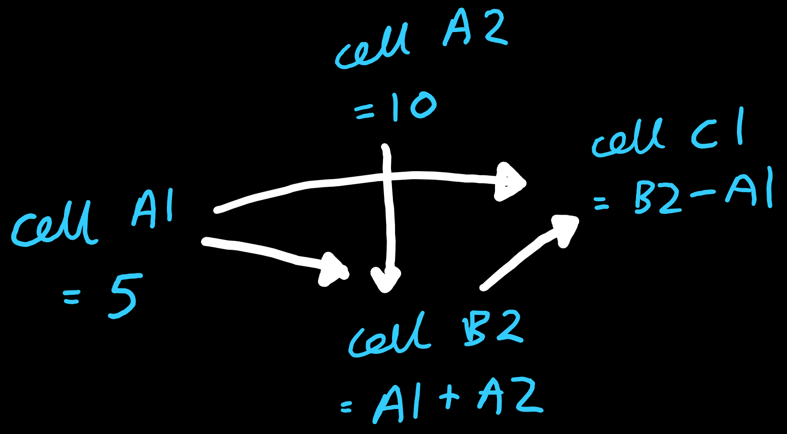
SCHEDULING PROBLEMS

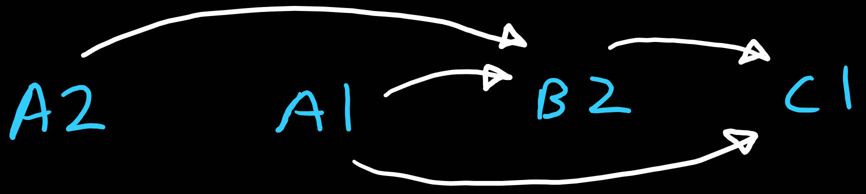
Let V be the set of tasks.

Let there be an edge $v \rightarrow w$ if v must be completed before w.

How can we arrange all the vertices into a sequence such that all edges point right?

For what graphs is this even possible?





Fundamentals of a method for evaluating rail net capacities, T.E.Harris and F.S.Ross, 1955

NOTICE: THIS DOCUMENT CONTAINS INFORMATION AFFECTING THE NATIONAL DEFENSE OF THE UNITED STATES WITHIN THE MEANING OF THE ESPIONAGE LAW, TITLE 18, U.S.C. SECTIONS 793 and 794. THE TRANSMISSION OR THE REVELATION OF ITS CONTENTS IN ANY MANNER TO AN UNAUTHORIZED PERSON IS PROHIBITED BY LAW.

SECRET RM-1573
10-24-55
-33-

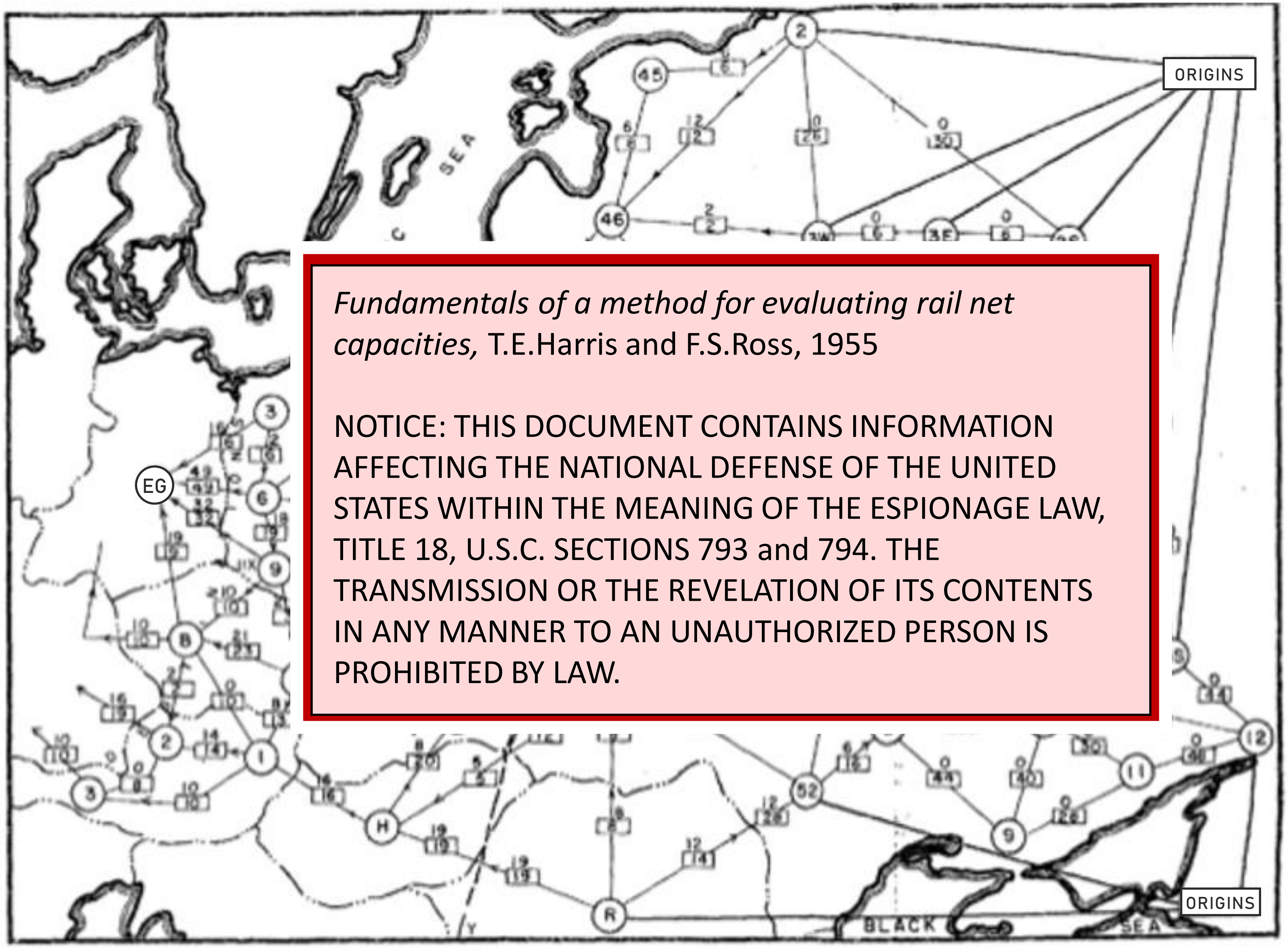


Fig. 7 — Traffic pattern: entire network available

Legend:

—·· — International boundary

(9) Railway operating division

$\xleftarrow{\overset{9}{\boxed{12}}}$ Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

All capacities in $\left.\begin{matrix}\text{trains}\\ \checkmark\text{1000's of tons}\end{matrix}\right\}$ each way per day

Origins: Divisions 2, 3W, 3E, 25, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovakia); and 2, 3 (Austria)

Alternative destinations: Germany or East Germany

Note IIX of Division 9, Poland

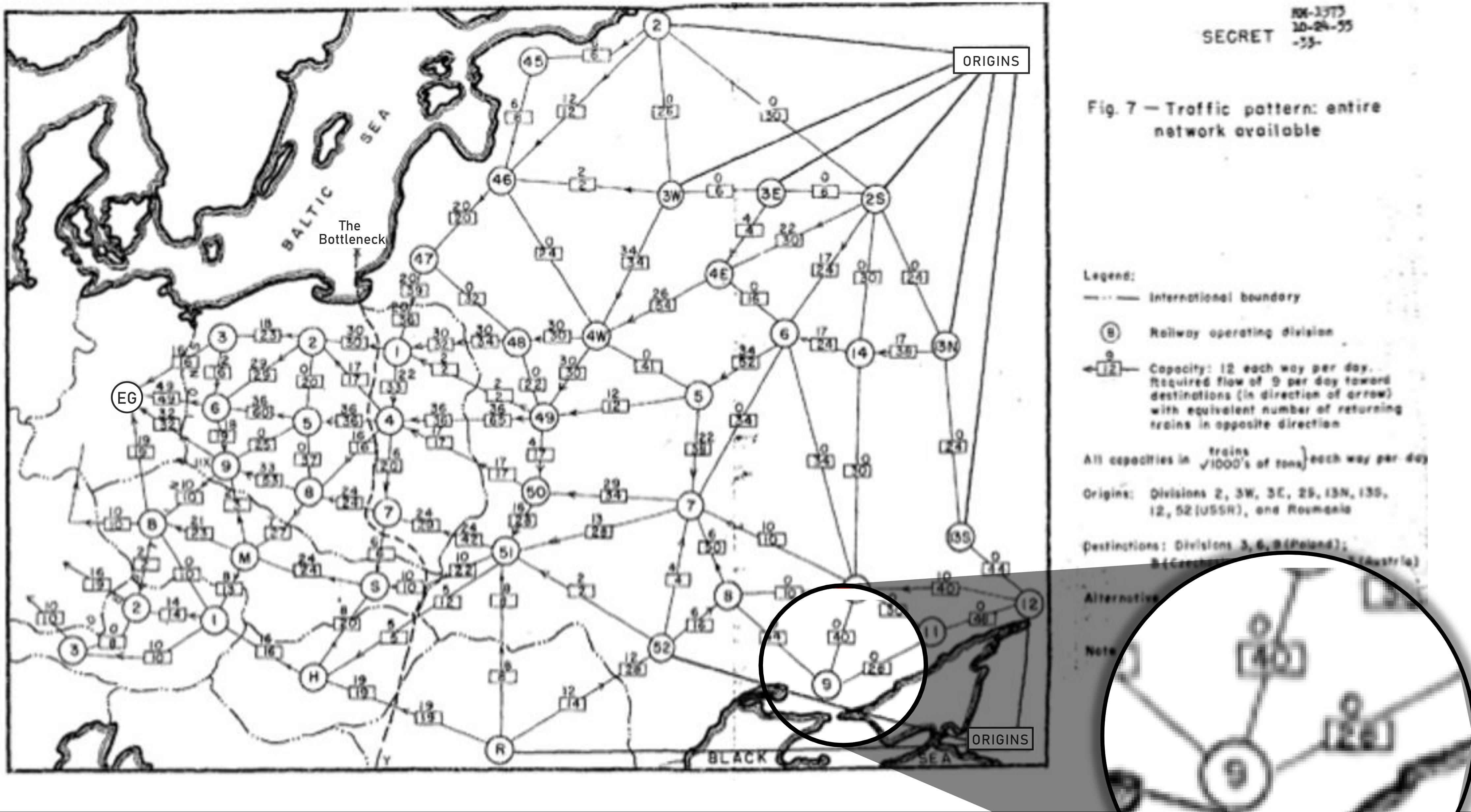

SECRET
RM-3573
10-24-55
-33-
ORIGINS
The Bottleneck
BALTIC SEA
BLACK SEA
EG
ORIGINS
Legend:
International boundary
Railway operating division
Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction
All capacities in trains / 1000's of tons each way per day
Origins: Divisions 2, 3W, 3E, 2S, 13N, 13S, 12, 52 (USSR), and Roumania
Destinations: Divisions 3, 6, 9 (Poland); B (Czech
(Austria)
Alternative
Note

Fig. 7 — Traffic pattern: entire network available

Each edge is labelled with its capacity (trains/day)

SECRET RM-1573 10-24-55 -33-

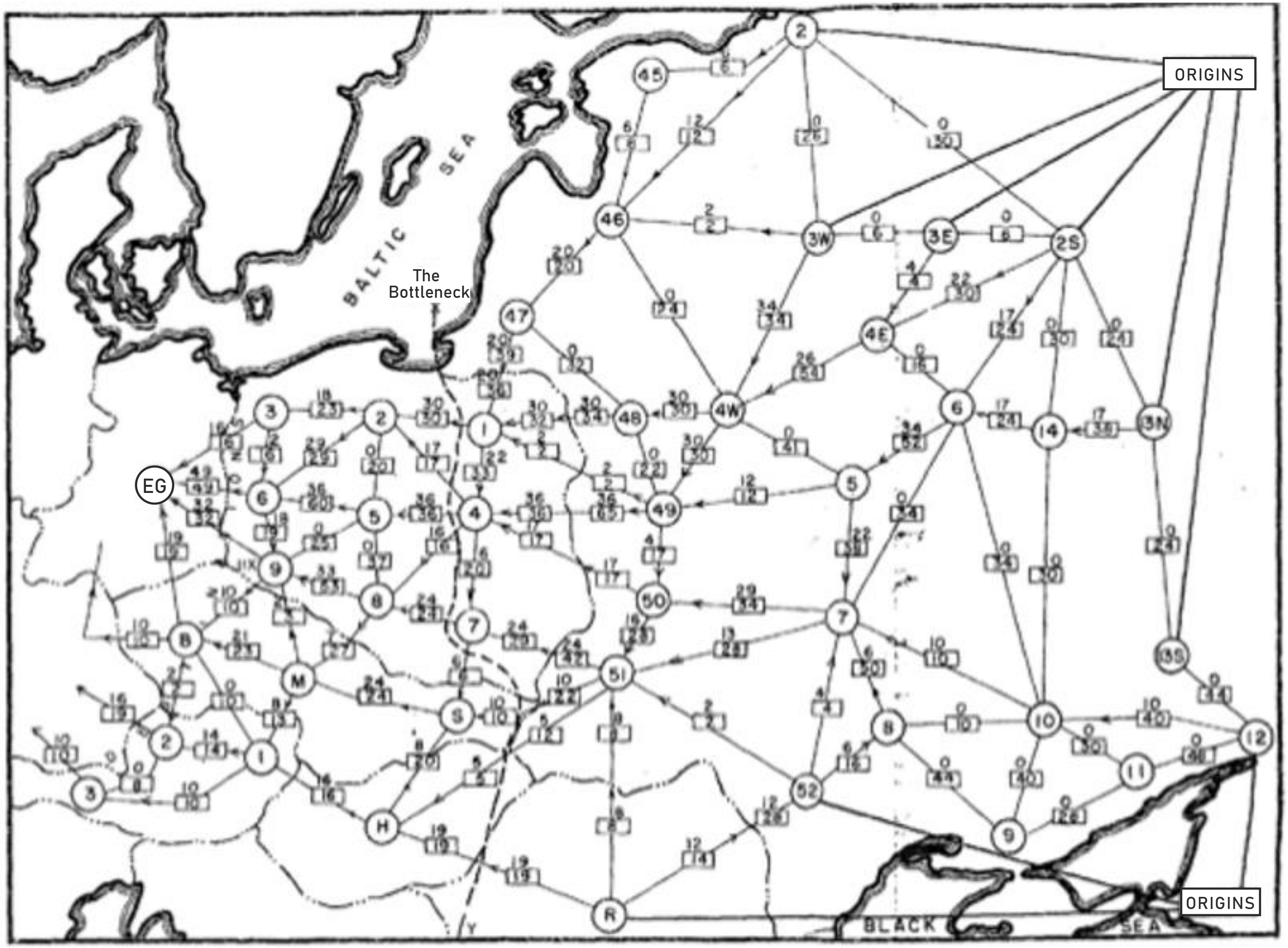


Fig. 7 — Traffic pattern: entire network available

Legend:

—··— International boundary

⑧ Railway operating division

←$\overset{9}{\boxed{12}}$— Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

All capacities in $\left.\begin{matrix}\text{trains}\\ \checkmark\text{1000's of tons}\end{matrix}\right\}$ each way per day

Origins: Divisions 2, 3W, 3E, 2S, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovovakia); and 2, 3 (Austria)

Alternative destinations: Germany or East Germany

Note IIX of Division 9, Poland

Q. What algorithmic questions might the US Air Force ask about this graph?

SECRET RM-1573
10-24-55
-33-

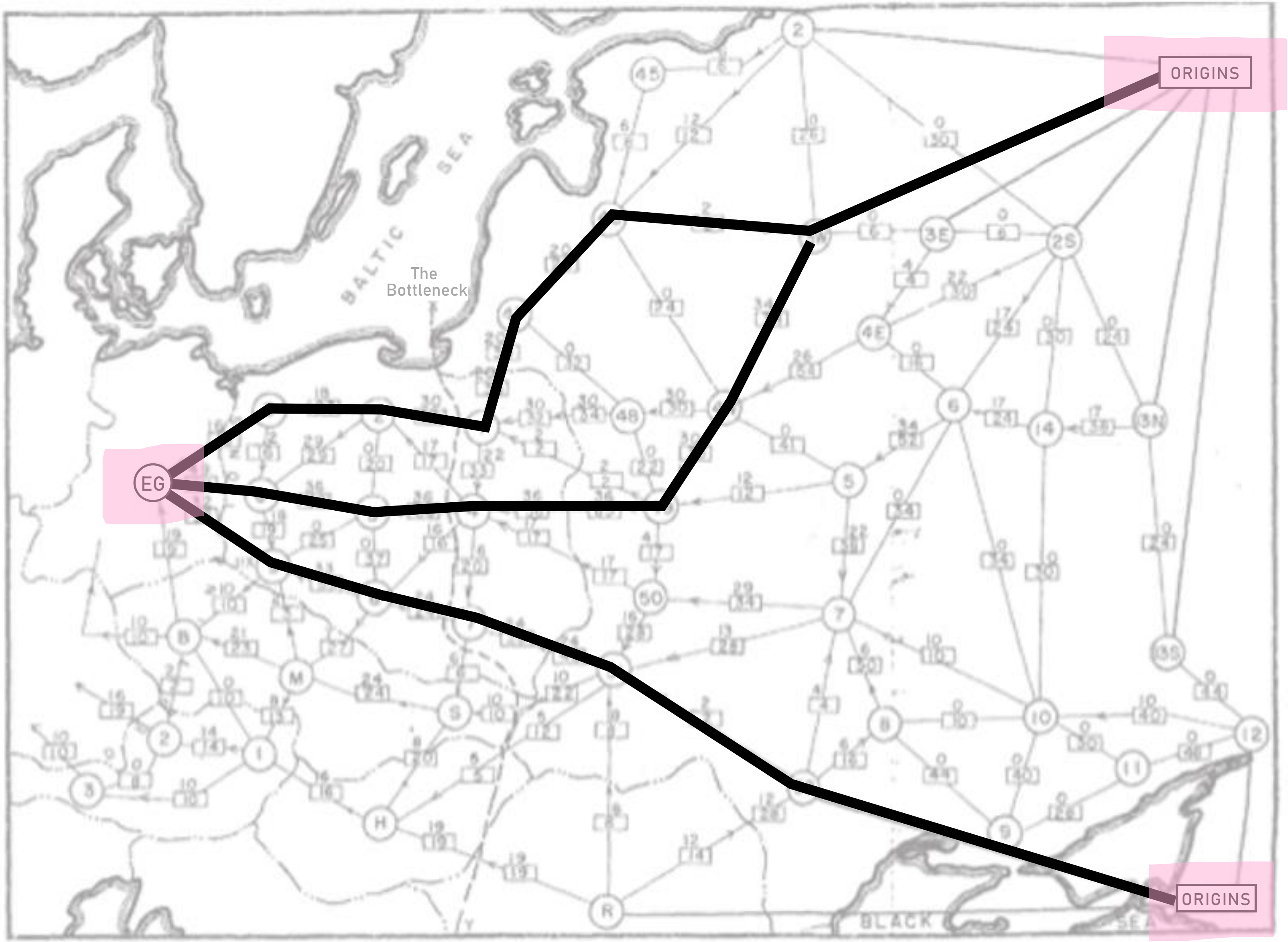


Fig. 7 — Traffic pattern: entire network available

Legend:

— · · — International boundary

⑧ Railway operating division

$\xleftarrow{\boxed{12}}$ with 9 above: Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

All capacities in $\left.\begin{matrix}\text{trains}\\ \sqrt{}\text{1000's of tons}\end{matrix}\right\}$ each way per day

Origins: Divisions 2, 3W, 3E, 2S, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); B (Czechoslovakia); and 2, 3 (Austria)

Alternative destinations: Germany or East Germany

Note IIK of Division 9, Poland

RM-1573
10-24-55
-53-

SECRET

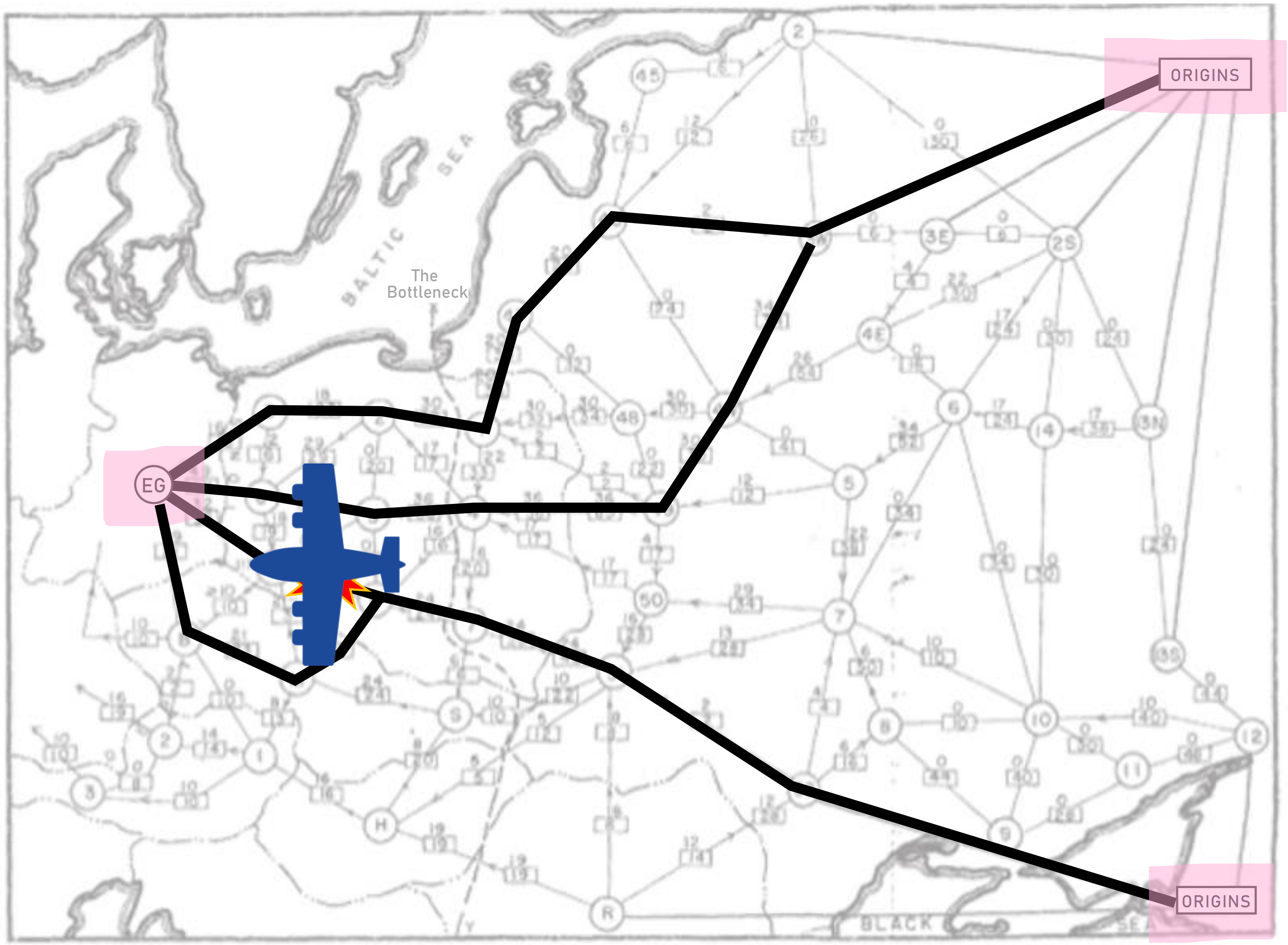


Fig. 7 — Traffic pattern: entire network available

Legend:

—·· — International boundary

⑧ Railway operating division

9
←[12]— Capacity: 12 each way per day. Required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

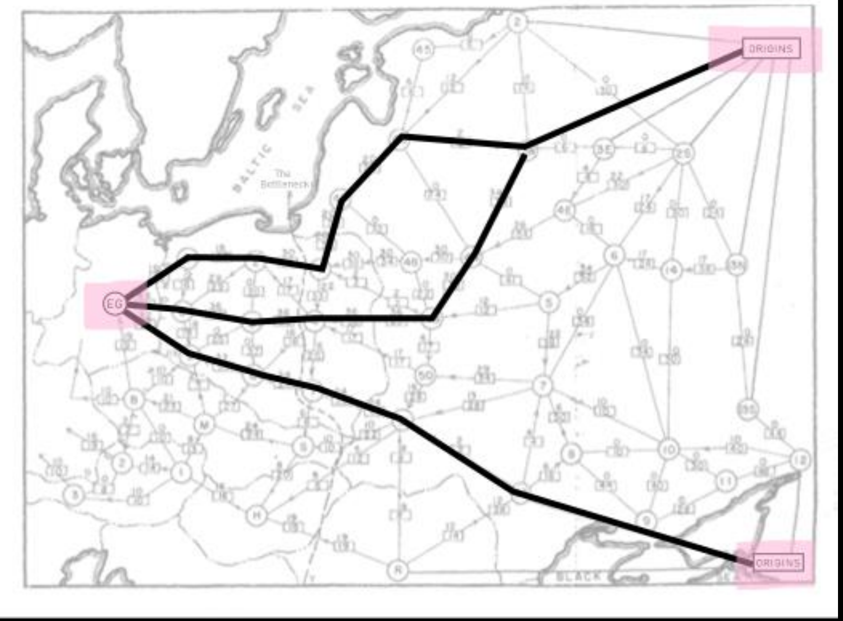
All capacities in trains / √1000's of tons } each way per day

Origins: Divisions 2, 3W, 3C, 2S, 13N, 13S, 12, 52 (USSR), and Roumania

Destinations: Divisions 3, 6, 9 (Poland); 8 (Czechoslovakia); and 2, 3 (Austria)

Alternative destinations: Germany or East Germany

Note IIK of Division 9, Poland



Flow networks

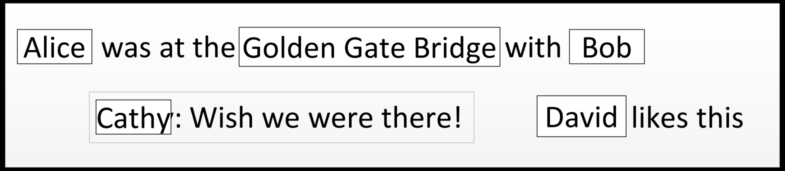
Consider a graph where each edge is labelled with a capacity $c: E \to \mathbb{R}$.

Let there be a source vertex s.

Let the *value* of a flow $f: E \to \mathbb{R}$ be

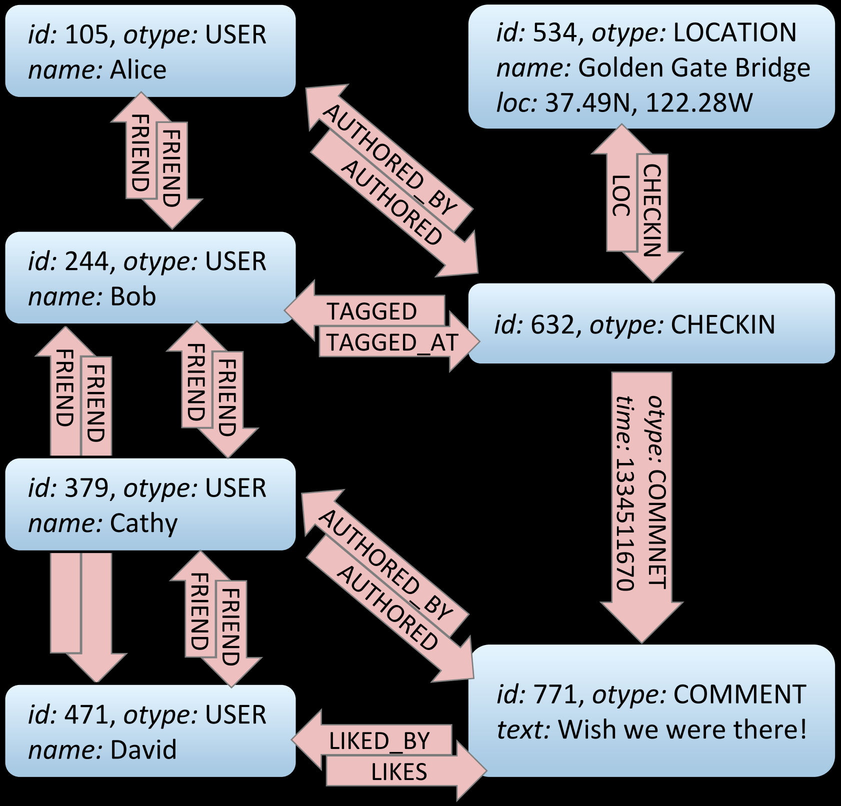
$$\text{value}(f) = \sum_{w: s \to w} f(s \to w)$$

- How can we find a flow of maximum value, given capacity constraints?
- Which are the edges whose removal would reduce the maximum flow value?

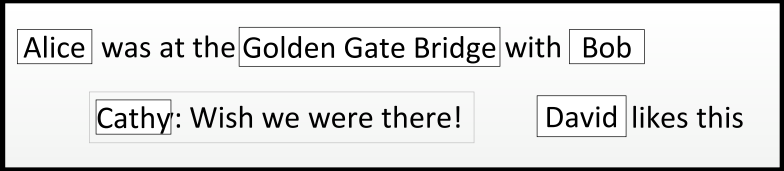


Alice was at the Golden Gate Bridge with Bob

Cathy: Wish we were there! David likes this



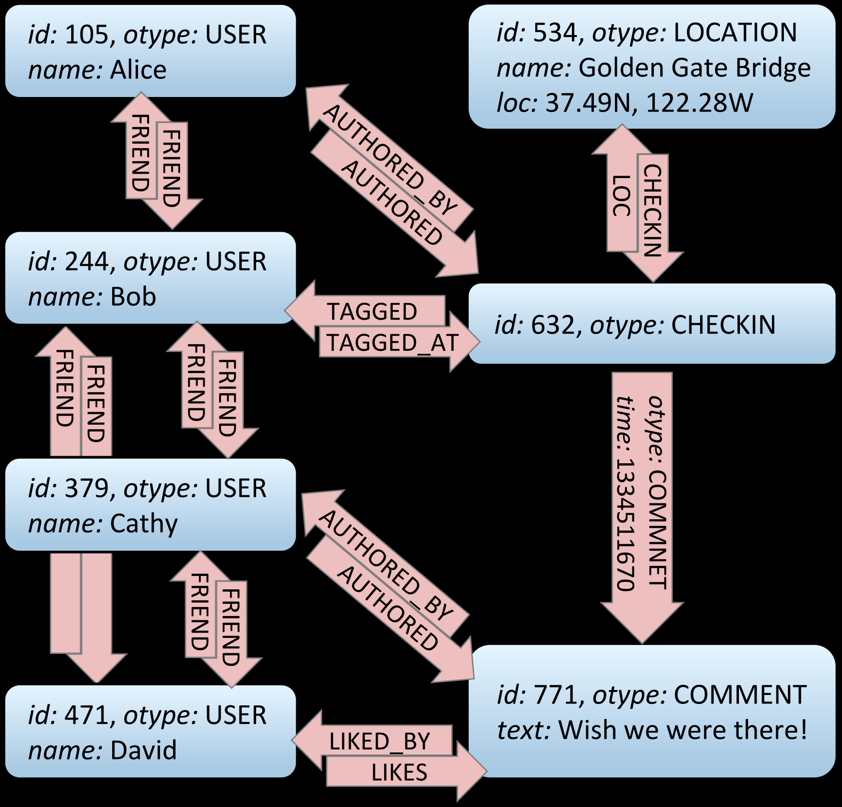
Q. Why did Facebook choose to make CHECKIN a vertex, rather than a USER→LOCATION edge?



Alice was at the Golden Gate Bridge with Bob

Cathy: Wish we were there!

David likes this



id: 105, *otype:* USER
name: Alice

id: 534, *otype:* LOCATION
name: Golden Gate Bridge
loc: 37.49N, 122.28W

id: 244, *otype:* USER
name: Bob

id: 632, *otype:* CHECKIN

id: 379, *otype:* USER
name: Cathy

id: 771, *otype:* COMMENT
text: Wish we were there!

id: 471, *otype:* USER
name: David

FRIEND
FRIEND
AUTHORED_BY
AUTHORED
LOC
CHECKIN
TAGGED
TAGGED_AT
FRIEND
FRIEND
FRIEND
FRIEND
otype: COMMINET
time: 1334511670
AUTHORED_BY
AUTHORED
FRIEND
FRIEND
LIKED_BY
LIKES

Q. What algorithmic questions we might ask about this graph?

What this course is about

- Clever algorithms
- Performance analysis
- Proving correctness

Topic for Fri 25 Feb

- Interplay between data and algorithm
- What we can model with graphs